Inleiding tot Matlab

Brecht Baeten

31 mei 2016

1 Inleiding

De broncode van deze tekst alsook alle voorbeelden kunnen worden gedownload via `https://github.com/BrechtBa/inleiding-tot-matlab/`.

2 Command line

Wanneer Matlab gestart is kan je rechtstreeks commando's uitvoeren in de command line interface. Typ hier bijvoorbeeld:

```
disp('Hello World!')
```

3 Scripts

Omdat het telkens opnieuw invoeren van commando's achter elkaar niet zo praktisch is, is het interessant om een reeks commando's op te slaan in een script en dit uit te voeren. Matlab code kan opgeslagen worden in tekst bestanden met de extentie ".m". Binnen de Matlab IDE zit een handige text editor specifiek ontworpen voor het schrijven van m-files. Open een de m-file editor en typ "`disp('Hello World!')`" en sla het bestand op als hello_world.m. Om het bestand uit te voeren kan je de naam van het bestand typen in de command line interface of op de run knop in de werkbalk van de editor klikken. Het resultaat is hetzelfde als daarnet.

Matlab heeft een aantal ingebouwde variabele types. Open een nieuw bestand, typ onderstaande commando's en sla het op als variables.m en voer het uit.

```
% float
A = 1;
A

B = 3.572;
B

% string
C = 'string';
C

% vector
```

```
D = [1,2,3];
D
D(1)
D(end)

% cell
E = {1,'test',4,D};
E

% structure
F = struct('value',1, 'name','test', 'spam','eggs');
F
F.spam
```

Alle reeds gedefinieerde variabelen zijn nu ook beschikbaar in de console. Typ bijvoorbeeld:

```
E(3)
```

of:

```
A+B
```

De grote kracht van Matlab zit in het werken met vectoren en matrices (Matlab komt van MATrix LABoratory). Vectoren kunnen enkel waarden van hetzelfde type bevatten. Een *Cell array* daarentegen kan waarden van een verschillend type bevatten. De Matlab *Struct* variabele is zeer interessant omdat deze kan gebruikt worden om verschillende soorten data gestructureerd te groeperen. In Matlab wordt elke variabele door zijn waarde gedefinieerd.

Scripts kunnen in Matlab alleen uitgevoerd worden indien ze in de Matlab current folder staan of als ze in een folder die tot het matlab zoekpad behoort staan. Met behulp van het *Current Folder* venster kan je binnen Matlab browsen naar een andere folder, dit kan ook met het "`cd()`" commando. Met behulp dan het "`addpath()`" commando met als argument een string kan je een bepaalde folder, waarin bijvoorbeeld een aantal veel voorkomende functies staan, aan het matlab path toevoegen. Zo worden deze functies beschikbaar.

Echt programmeren begint pas bij maken van lussen en voorwaarden: flow control. In Matlab is dit zeer eenvoudig met behulp van "`for`" lussen of "`if else`" structuren:

```
for i=1:10
    if mod(i,2) == 0
        sprintf('%i is even',i);
    else
        sprintf('%i is oneven',i);

    end
end
```

In Matlab moet elke "`for`" lussen of "`if else`" constructie eindigen op een "`end`" commando. Het "`1:10`" commando in de "`for`" lus maakt een *Vector* met waarden van 0 tot en met 9. De "`sprintf('%f is even',i)`" functie formatteert zijn argumenten op de plaats van de "`%f`" in de format string al floating point numbers. Er zijn verschillende mogelijkheden om getallen te formateren, even zoeken in de help functie leert je heel veel over zulke ingebouwde functies.

Over arrays en structures itereren kan ook in Matlab. Om dit te doen defineer je een tijdelijke index variabel waarover je itereert, en je roept het element hodende bij die index op:

```
% structures
A = struct('foo', 'bar',
           'spam', 'eggs',
           'bacon', 'spam');

fields = fieldnames(A);
for i = 1:length(fields)
    sprintf('%s: %s', fields{i}, A.(fields{i}));
end

% looping over an array
B = [1,11,111,1111];
for i = 1:length(B)
    sprintf('B(%d) = %.0f',i,B(i))  ;
end

% simultaneous looping over arrays
C = [1,2,3];
D = [7,8,9];
for i = 1:length(C)
    sprintf('c = %0.0f, d = %0.3f', C(i), D(i));
end
```

4 Functies

Een set commando's die vaak op dezelfde manier gebruikt worden kan je groeperen in een functie. Een functie definieer je met het "**function**" keyword, gevolgd door de output argumenten, een gelijkheidsteken, de functie naam en de argument namen tussen haakjes op het einde van een functie hoort zoals gewoonlijk een "**end**". Functies moeten binnen Matlab in een apparte m-file met dezelfde naam als de functie worden opgeslagen.

```
function val = spam(A,B,C,D)
    % returns a a number as if the arguments were different digits in the
       number
    %
    % Parameters:
    % A: float, hundreds
    % B: float, decades
    % C: float, units
    % D: float, tenths

    val = 100*A+10*B+C+0.1*D;
end
```

De functie kan nu worden gebruikt als "`A = spam(1,2,3,4)`"Ṫariabelen die binnen een functie gedefinieerd zijn bestaan ook enkel in de functie namespace. Onderstaande code geeft dus een error omdat `val` enkel in de functie namespace gedefinieerd is, niet in de globale namespace. Wanneer de onderstaande code runt krijg je dan ook een foutmelding. Matlab geeft wel meteen aan waar de fout is opgetreden wat handig is om code te debuggen.

```
A = spam(4,1,5.3);
disp(A);
disp(val);
```

```
error: 'val' undefined
```

Onmiddellijk na de definitie volgt de documentatie van de functie als commentaar "`%`". Deze wordt weergegeven indien de de "**`help`** `spam`" functie wordt opgeroepen. Het is een goede gewoonte om hier steeds te schijven wat de functie doet, wat ze returnt, wat de argumenten betekenen en welk datatype de argumenten moeten zijn.

Matlab beschikt over een zeer groot arsenaal van ingebouwde functies die rechtstreeks beschikbaar zijn. Deze zijn ook zeer goed gedocumenteerd in de help functie die kan opgeroepen woorden door op *F1* te duwen. De ingebouwde zoekfunctie werkt ook zeer goed.

5 Vectoren en plotten

In Matlab kunnen vectoren met behulp van ingebouwde functies gecreëerd worden. Je kan elk element van een vector appart definieren, om de performantie te vergroten is het wel belangrijk dat je op voorhand de vector initializeerd met bijvoorbeeld allemaal nullen. Veel functies in Matlab kunnen echter ook met vectoren werken.

Functies om vectoren te plotten zijn ook ingebouwd in Matlab. Matlab kan ook gebruik maken van LaTeX om aslabels en legendes weer te geven. In het onderstaande voorbeeld worden een aantal vectoren opgebouwd en geplot.

```
x = linspace(0,2*pi,50);

% build an array elementwise
c = zeros(size(x));
for i = 1:length(x)
    c(i) = cos(x(i));
end

% vectorized functions
s = sin(x);

ss = cumtrapz(x,c);


% plotting
figure('Position', [100, 100, 400, 280]);
hold on;
plot(x,c,'b');
plot(x,s,'g-s');
plot(x,ss,'r',linewidth,2);
xlabel('$x$ (rad)','Interpreter','latex');
ylabel('$y$','Interpreter','latex');
legend({'cosinus','sinus','$\int$ cosinus(x) d$x$'},'Interpreter','latex');

print('sinus_cosinus','-dpdf')
```

Figuur 1: Plot voorbeeld